

# Perforated Backpropagation: A Neuroscience Inspired Extension to Artificial Neural Networks

Rorry Brenner, Laurent Itti  
Computer Science and Neuroscience,  
University of Southern California  
Perforated AI Inc.  
[rorry@perforatedai.com](mailto:rorry@perforatedai.com), [itti@usc.edu](mailto:itti@usc.edu)

**Abstract.** The neurons of artificial neural networks were originally invented when much less was known about biological neurons than is known today. Our work explores a modification to the core neuron unit to make it more parallel to a biological neuron. The modification is made with the knowledge that biological dendrites are not simply passive activation funnels, but also compute complex non-linear functions as they transmit activation to the cell body. The paper explores a novel system of "Perforated" backpropagation empowering the artificial neurons of deep neural networks to achieve better performance coding for the same features they coded for in the original architecture. After an initial network training phase, additional "Dendrite Nodes" are added to the network and separately trained with a different objective: to correlate their output with the remaining error of the original neurons. The trained Dendrite Nodes are then frozen, and the original neurons are further trained, now taking into account the additional error signals provided by the Dendrite Nodes. The cycle of training the original neurons and then adding and training Dendrite Nodes can be repeated several times until satisfactory performance is achieved. Our algorithm was successfully added to modern state-of-the-art PyTorch networks across multiple domains, improving upon original accuracies and allowing for significant model compression without a loss in accuracy.

**Keywords:** Artificial Neural Networks, Deep Learning, Speech Processing, Drug Discovery, Stock Forecasting, Machine Learning, Dendritic Integration, Cascade Correlation, Artificial Neurogenesis

## 1 Introduction

Advances are continuously being worked on in deep learning. After the large resurgence of neural nets for computer vision with the success of AlexNet (Krizhevsky, et al., 2012) many tweaks have been made to the system of connections between layers. From Google's Inception Modules (Szegedy, et al., 2015) to Microsoft's ResNets (He, et al., 2016) to modern image transformers (Parmar, 2018.) the results on the ImageNet object detection challenge (Russakovsky, et al., 2015) are improved every year.

The first mathematical model of an artificial neuron was invented in 1943 (McCulloch, 1943) based on 1943 neuroscience. This idea was brought into artificial intelligence via the perceptron in 1958 (Rosenblatt, 1958). Backpropagation through layered networks was added in 1970 (Linnainmaa, 1970). Since then, computer scientists have innovated new activation functions, optimizers, schedulers, and architectures, all while the hardware exponentially improved to bring us to where we are today. However, the neuron unit and the core backpropagation algorithm based on gradient descent have remained unchanged since their original invention. This paper explores the modern neuroscience understanding of active dendrites (Major, et al., 2013) and how such a concept could be added to an artificial neural network.

When exploring how to best instantiate this method, two restrictions were set. First, when active artificial dendrites were created, they must be outside of the network despite being able to affect individual neurons within the network. Second, something about them must operate differently than traditional artificial neurons. The Perforated Backpropagation paradigm achieves both goals. Critical to note, Perforated Backpropagation is an addition, and not a replacement. This means that it is not a novel architecture which would replace the work others have done, e.g., using this instead of a transformer

model or a ResNet. Perforated Backpropagation is a plug-in that functioning systems can quickly add, rather than a method that would require researchers to replace any other component of what makes their systems unique and successful or perform any major overhauls of their processes. By adding Dendrite Nodes to the neurons of state-of-the-art systems, Perforated Backpropagation is able to push the state of the art even further by keeping the same overall architectures but making each of the neuron nodes better at their original roles.

## 2 Background

### 2.1 Biological Background: Active Dendrites

The human brain is an amazing computational machine. Its visual system has been said to be the only evidence that generalized object recognition is possible. This is why artificial neurons follow its design in a computational parallel. However, the original artificial neuron is a significantly reduced version of a biological neuron. This artificial neuron is still highly functional because to capture all of the functional capabilities of a biological neuron, an artificial neuron does not need to model every single atom. The question becomes, not only what functions are missing in artificial neural networks that biological networks perform, but also what functions are essential for their aptitude. The function to inspire this work is that of active dendrites.

In modern state-of-the-art artificial neural networks, dendrites are nonexistent (Krizhevsky, et al., 2012; Szegedy, et al., 2015; Ciregan, et al., 2012; Kubilius, et al., 2016). Presynaptic neurons form synapses directly to the cell bodies of postsynaptic neurons with connection weight as their only parameter (Widrow & Lehr, 1990). These algorithms are not at a place where they can compete with the human visual system. Artificial networks differ in many ways from those of biology, but one certain advantage the human visual system has is the increased computational power within a single neuron by way of their dendrites.

The simplest function dendrites perform is physically being the structure on which most synapses form to collect presynaptic activation and transport it to the soma (Kandel, et al., 2000). Even this most basic function still behaves in a way that must be considered. Dendrites are not perfect cables, they are leaky. Voltage attenuates significantly if it travels down the dendrite with no further external influence (Stuart & Spruston, 1998). The axial resistance is lower towards the soma so voltage will flow towards the cell body before leaking out through the membrane. Collecting voltage for the cell is only a fraction of their complete ability. Passive summation over dendritic voltages plays an important role. Coinciding peak voltages cause a greater surge, more likely to cause a spike, at the soma. This allows the neuron a type of temporal calculation of synapse firing order which might only peak if inputs are in a proximal-to-distal order (Rall, 1964; Koch, et al., 1983). Another important calculation dendrites provide, and the one we will discuss adding to artificial neurons, is their active ability to generate spikes. The exact details of this ability are still disputed by some scientists. A paper by Major et al. shows a framework in which conflicting results can all fit (Major, et al., 2013).

NMDA is the dominant spike type in dendrites. NMDA allows both thresholded and graded electrogenesis. Depending on glutamate concentration from synaptic firing, NMDAR conductance can be up-stable, down-stable, or unstable. Unstable is the condition which allows for a thresholded response. NMDAR conductance has a unique N-shaped I-V curve as seen in Figure 1. This N shape can have one early intercept, three intercepts, or one later intercept. At low glutamate concentrations the trough of the curve representing current flow is completely above the axis meaning current flows outward at typical biological membrane potentials. As the concentration increases the trough gets deeper until it hits a point where it dips below zero. This causes the curve to cross the axis at three points. The middle crossing is an unstable

state for voltage. If the instantaneous voltage is below this intercept voltage will flow outward, lowering the membrane potential until it reaches the first intercept, which is a stable state. If voltage is allowed to surpass this point through presynaptic firing the current is now inward, causing the membrane to further increase up to the third and final intercept, another stable state. Passing this middle intercept will cause a spike. This condition, with the existence of the unstable intercept, is the only one of the three where a spike can happen. At extremely high glutamate concentrations the entire N is below the axis causing current to flow inward until reaching the last intercept of the curve. As glutamate concentration decreases this effect happens in reverse and the high-voltage stable state ceases to exist. This causes the voltage to quickly fall to the low voltage stable state, ending the spike.

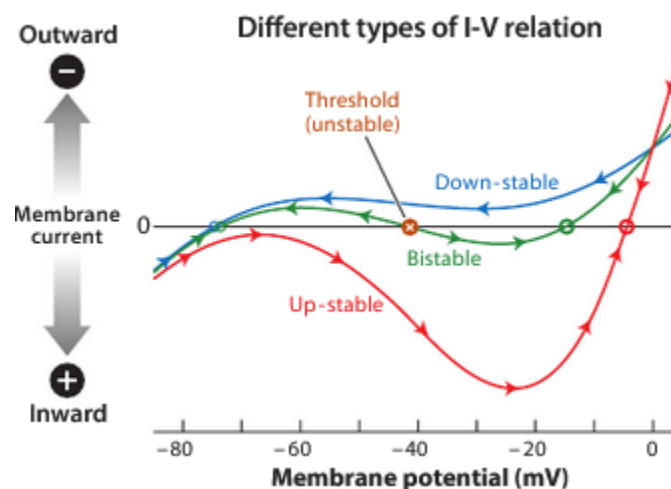


Figure 1. Graph of three I-V curves at different glutamate concentrations. Blue shows low concentration where the current flows out, red shows high concentration where current flows in, green shows intermediary concentrations where a spike can occur. Image from (Major, et al., 2013).

An NMDAR I-V curve being based on glutamate concentration causes an interesting effect. Unlike the voltage dependent spiking of an action potential, NMDA spikes are still voltage dependent, but the exact voltage can change. It is still all-or-none, in the sense that there is a voltage threshold above which the voltage will spike to a higher value. This threshold, however, does not exist at low glutamate levels, slowly lowers as concentration increases, and then ceases to exist again at high concentrations (Major, et al., 2013). The threshold and amplitude of the spike both increase from distal to proximal dendritic locations in correlation with the respective input conductance. Synaptic firing, on the other hand, has little effect on the amplitude of the spike, and only serves to increase the duration of the spike as glutamate release continues (Major, et al., 2008). Clearly, NMDA receptors are already dynamic on their own, but this nonlinear function only gets more complex when considering a system with multiple connections. All of this discussed functionality is missing from current artificial neural network models. This significantly

impedes the potential of artificial neural network performance, as some neurobiological research supports the idea that dendrites, and not neurons, are the fundamental functional unit of the brain (Branco, 2010).

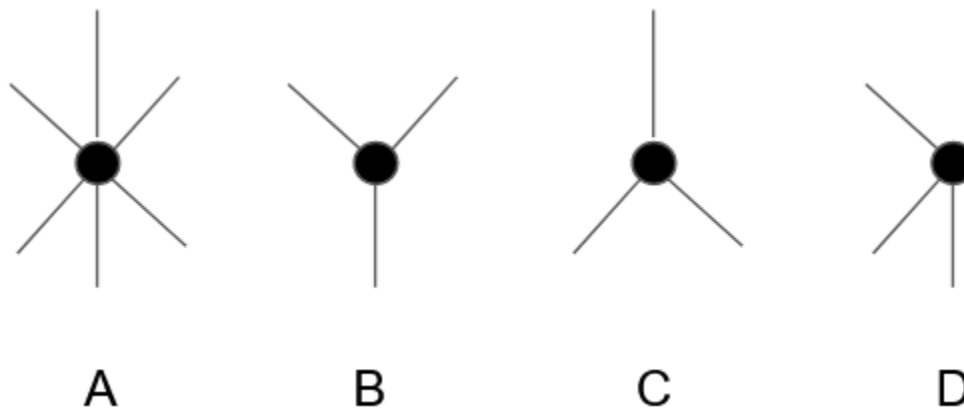


Figure 2. Four examples of connected lines. B and C are y-vertices, D is an arrow-vertex, and A is six lines coming together at a point.

This ability of biological dendrites allows individual biological neurons to code for features that would require multiple neurons in more than one layer in an artificial network. An easy to visualize example comes from the non-accidental properties of Biederman's geon theory of shape perception (Biederman, n.d.). Figure 2 shows an example of four vertices. B and C are y-vertices, while D is an arrow-vertex and A is 6 lines converging on a point. Our visual system may naturally associate distinct partial three-dimensional shapes to some of these: B and C might be perceived as a corner facing the observer, while D is a corner facing away, and A is an assortment of lines. Individual neuronal recordings have shown that, in biological visual systems, neurons are sensitive to these types of non-accidental properties when looking at objects (Kayaert, et al., 2003). A single artificial neuron could not code for a feature like this because if the lines provide positive input, B and C could not cause high activation while A and D do not. The motivation to add artificial dendrites to the neurons of an artificial network is to allow individual neurons to learn more complex features such as these.

As artificial neurons do not replicate the exact function of biological neurons, the Dendrite Nodes of Perforated Backpropagation do not replicate the exact function of NMDA spikes. But similar in function to biological dendrites, this work explores if better systems can be created by granting individual neurons the ability to make more complex decisions by adding nonlinear processing between each neuron and the presynaptic neurons from the previous layer. However, this nonlinear processing is added in a different way than the spline method of the recently published paper on Kolmogorov-Arnold Networks (Liu, 2024). To simplify this system and allow it to be compatible with modern deep learning frameworks, here additional tensor processing layers are used as a proxy for artificial computation-enabled dendrites. However, a method is devised to ensure that this particular subset of layers is outside of the network, by adjusting the way they interface with traditional backpropagation of modern deep learning systems. This way, we propose a new network design and information processing paradigm that is distinct from adding more neurons and connections to a standard deep neural network.

## 2.2 Computational Background: Cascade Correlation

One of the restrictions set for this research was ensuring Dendrite Nodes behaved or learned differently than neuron nodes. To that end, a starting point was chosen from an earlier algorithm which also instantiates a network that has nodes which operate differently from each other. Cascade Correlation (Fahlman & Lebiere, 1989) is a system developed to improve single-layer networks. It utilizes artificial neurogenesis to continuously improve error by adding new hidden nodes between the input and output layers. Learning takes place by alternating between two phases: typical perceptron learning, and cascade node learning. During perceptron learning, each output node learns as a standard perceptron to minimize its error based on its inputs. During cascade node learning, first, the weights of the network are frozen. Next a set of candidate nodes are created which are not a part of the network yet. They are given inputs from the same input nodes as the perceptrons of the output layer, but they form no output connections. To determine their input weights, they use the total error of the output nodes in their loss function. When given a test point, if the total error of the system is high the new nodes strengthen their connections, and if the total error is low, they weaken them. With this process new nodes learn to maximize their correlation with the error of the network.

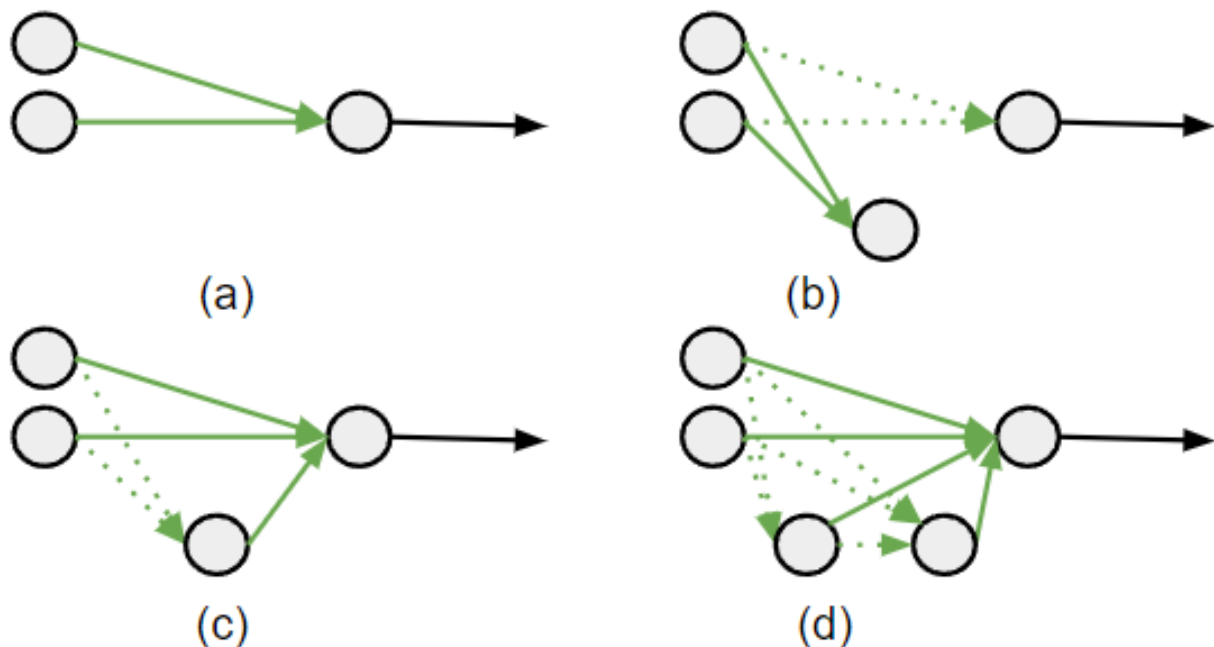


Figure 3. Stages of Cascade Correlation learning are shown. In [a] a perceptron is shown with 2 inputs and 1 output. In the first phase this perceptron is trained to maximize classification of the input data. Next in [b] the weights from the input to the network are frozen, signified by dotted lines, and the new node learns weights which maximize its correlation with the perceptron's error. [c] shows the network in the next phase with the new node added after having its weights frozen and the perceptron learning to modify its input weights as normal with the one additional input from the new node. In [d] an additional node has been added with connections from both the input and the other created node.

Once training is complete the candidate node which achieved the best correlation score is added to the network. This is done first by fixing its input weights, which are never changed again, forming a permanent snapshot of the network's error at the timepoint it was added. Next the candidate node forms output connections to each of the output nodes. Then the candidate training phase is over, switching back to perceptron learning where each output node treats their new input connection identical to their connections directly from the input. The only additional complexity is that during the next phase when new candidate nodes are added they also form input connections from all previously created cascade nodes. Figures 3 and 4 show visuals to explain the architecture itself, and what the architecture can do. For simplicity they both use two inputs to represent x and y coordinates, making the perceptron a linear classifier.

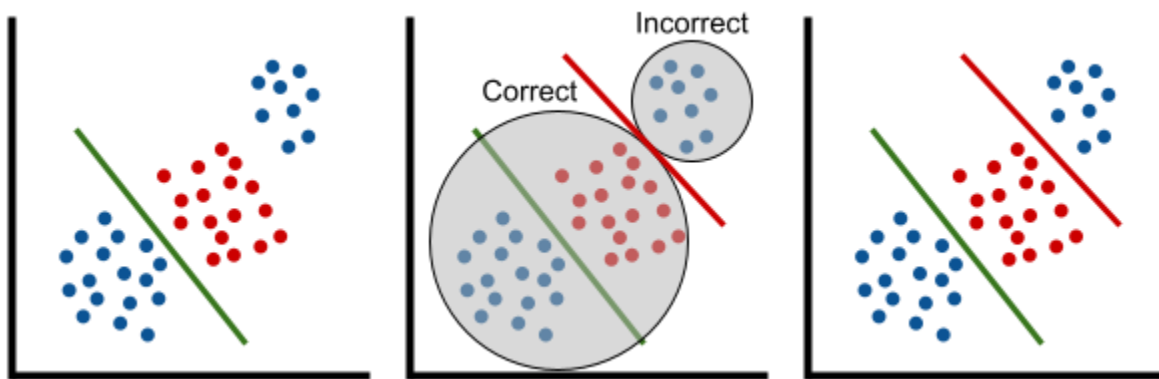


Figure 4 These graphs show the linear classification capabilities of a Cascade Correlation network. On the left a single perceptron can only create a single classification line. For simplicity, this line has been drawn to perfectly classify the blue data on the left and all the red data, while getting its classification wrong on all the blue data on the right. The center image shows how the Cascade Correlation node learns. Rather than classifying the data, it learns to classify where the perceptron is wrong. If there were more than one output node it would learn to classify where all of the outputs make the most mistakes. On the right the red line represents the classification line of the new node. The output perceptron now has not only the x and y coordinates as input but also an input representing if those points are above or below the red line. It can now modify its three weights to correctly classify all of the data.

### 3 Related Work

The concept of adding artificial dendrites to machine learning models of artificial neurons has been around since 2003 (Ritter, 2003). Early work focused on morphological perceptron models while modern works also leverage traditional artificial neuron models. Traditional neural networks use a weighed sum methodology leading into a nonlinearity. Morphological perceptron models instead use lattice algebra including maximum and minimum functions. Works related to artificial neuron models with dendrites mostly focus on shallow networks with only a single layer of neurons and any additional layers composed of dendrites. Because they use shallow networks, they tend to focus on efficiency rather than accuracy. A review paper from 2021 (Chavlis, 2021) found that none of the dendrite models they reviewed achieved state-of-the-art accuracy on the datasets they were applied to. However, they did find the models achieved better performance compared to traditional networks with the same number of parameters.

From the morphological subset one group of authors (Sossa, 2014) introduces the additional of dendrites to morphological perceptrons which can train dendrites to perfectly classify any training dataset. The algorithm instantiates a neuron with dendrites for a set of input data by iteratively creating hypercubes which encapsulate possible input patterns. First a hypercube is generated which encapsulates all the data. Then if the training data inside a hypercube is entirely within a single class the hypercube is saved. If the training data is not entirely within a single class, the hypercube is iteratively split into smaller hypercubes. Once this process is complete the neuron is given a set of dendritic branches corresponding to each hypercube where the branches have frozen weights such that a particular branch is only active if a datapoint is within its associated hypercube. The authors showed this algorithm could perform well on basic pattern recognition and even be applied to image processing tasks outperforming simple multi-layer perceptrons. Similar work showed this concept could be extended to include k-means++ clustering for hypercube initialization (Arce, 2018).

Research has also been performed which sets up architectures that are inspired by dendrites, but follows traditional neural network theory by using standard backpropagation for training. One set of authors (Li, 2020) was able to show this architecture could achieve improved performance with significantly reduced power consumption from hardware. Another paper (Meir, 2023) explored a dendritic structure applied to cifar-10 (Krizhevsky, 2009) which outperformed the larger LeNet (LeCun, 1998). However, these experiments were performed comparing the methods to other small models and not state-of-the-art architectures.

In addition, Artificial dendrite research has been performed which uses traditional neurons but strays from traditional neural network training by not using backpropagation as its training paradigm (Gao, 2018). Instead, the authors ran experiments using six learning algorithms: biogeography-based optimization, particle swarm optimization, genetic algorithm, ant colony optimization, evolutionary strategy, and population-based incremental learning. While the original authors only compared this method to multi-layer perceptrons, further research (Egrioglu, 2022) showed this algorithm to be the best machine learning method for a time series forecasting problem using the particle swarm method, though it did not outperform algorithms which did not use machine learning. Another group of researchers (Guerguiev, 2017) used the dendrites themselves to develop an alternative error signaling that allows error to be transmitted from the output layer to hidden layers with a more biological method than traditional backpropagation. This system used “basal” compartments for forward processing of input data and “apical” compartments for backwards processing of output error.

A key difference between our method and the discussed research is that other dendrite methods fundamentally change the network or training paradigm. For a modern machine learning researcher to switch from a standard neural network to a neural network with dendrites these methods would at a minimum require adjustment of the definitions of their models to include additional sparse layers, and in most cases require a fundamental shift away from the training process they currently use to switch to the new paradigm. One of the advantages of Perforated Backpropagation is that it is not a shift away from the norm that requires significant changes to modern methods. Instead, it is a layer that can be added on top of standard deep learning pipelines that requires no adjustment to model definitions, training parameters, or datasets. With Perforated Backpropagation model is wrapped in a way that allows the existing neurons within tensor based deep neural networks to be optimized with Dendrite Nodes.

## 4 Perforated Backpropagation

In this section original network nodes are referred to as neurons and Cascade Correlation nodes as Dendrite Nodes. Similar to the process in Cascade Correlation, during Perforated Backpropagation the network is first fully trained with traditional backpropagation until no more improvements to training error are made with additional epochs through the training data. At this point every neuron in the network is given a set of associated candidate Dendrite Nodes, which behave in the same way as the candidate nodes of a Cascade Correlation system. All the math from Cascade Correlation learning transfers over with only a minor modification. The difference to consider is that the original Cascade Correlation system was only created to work with perceptrons, i.e. networks where the original architecture had no hidden layers at all. To apply this methodology to deep learning a change that must be made is adding Dendrite Nodes to deeper neurons in the hidden layers of the network. The problem of how their Dendrite Nodes learn is solved by relying on the error method already in use, backpropagation. However, the key contribution of this paper is the development of the novel "Perforated" Backpropagation paradigm.

During Perforated Backpropagation, error from the output is propagated through the network, but the calculations involved in the gradient descent equations only calculate error terms for neurons and only factor weights and output values from neuron-to-neuron connections, completely ignoring connections going through Dendrites. However, the error gradient calculated at each neuron is not only used to update its weights during normal learning, but during Dendrite Node learning the Dendrite Nodes learn to maximally correlate their output activation with the backpropagated error of the single neuron they are connected to. This training will continue until no neuron has a Dendrite Node which is still improving its correlation. Perforated Backpropagation allows the network to both stay the same and be added to. By removing the Dendrite Nodes from the gradient descent equations, the model continues to have the same network architecture as defined by the error communication. This allows each neuron in a network to continue to code for the same feature, and have the same role, as it did in the original network. However, each neuron is also empowered to be better at coding for that feature by perforating its calculated error to one or more Dendrite Nodes that can learn to identify patterns where the neuron is making mistakes and then adjust the weight of its connection to those Dendrite Nodes to make better decisions about these outliers.

Traditional backpropagation through a multilayer perceptron uses the equation from (Norvig, 2003) with the following definitions.  $\Delta_j$  is the error associated with neuron node  $j$ .  $g'(in_j)$  is the derivative of the activation function  $g$  applied to the total input to neuron node  $j$ . Finally,  $\sum_i W_{i,j} \Delta_i$  is the sum of each weight between postsynaptic neuron node  $i$  and presynaptic neuron node  $j$  multiplied by the error associated with neuron node  $i$ .

$$\Delta_j = g'(in_j) \sum_i W_{i,j} \Delta_i \quad (3.1)$$

For Perforated Backpropagation, this equation is modified such that postsynaptic nodes are split into postsynaptic neuron nodes and postsynaptic Dendrite Nodes. With postsynaptic neuron nodes designated as  $i$ , and post synaptic Dendrite Nodes designated as  $k$ , the equation is split across the summation with the Dendrite Node error terms zeroed out during the backpropagation process.



$$\Delta_j = g'(in_j)(\sum_i W_{i,j}\Delta_i + 0 * \sum_k W_{k,j}\Delta_k) \quad (3.2)$$

Additionally, with Perforated Backpropagation  $\Delta_k$  can be calculated differently. For these first experiments,  $\Delta_k$  was chosen with a slight modification to the original Cascade Correlation formula which is originally taken from the formula for covariance of two variables. This modification was made because the original formula takes multiple output neurons into account, as well as expecting significantly smaller datasets. In our case the following formula is applied during each mini-batch with the following definitions.  $g(in_k)$  is the activation function of the Dendrite Node applied to the sum of its input.  $\bar{g}(in_k)$  is a running average over previous activations of the Dendrite Node. Similarly,  $\bar{\Delta}_i$  is a running average of the errors of the associated neuron node.

$$\Delta_k = (g(in_k) - \bar{g}(in_k))(\Delta_i - \bar{\Delta}_i) \quad (3.3)$$

The weight update rule for input connections remains largely the same as the original paper with parallel changes given the updated  $\Delta_k$ . The new symbol  $\sigma$  in this formula is the sign of the average correlation between the Dendrite Node's output and the associated neuron node's output.

$$\delta\Delta_k/\delta w_j = \sigma(\Delta_i - \bar{\Delta}_i) g'(in_k) \quad (3.4)$$

Rather than choosing a single candidate node to add to the network, as in Cascade Correlation, every neuron adds its associated candidate Dendrite Node with the highest correlation to its error. After adding the candidate Dendrite Nodes, all their input weights are locked. At this point every neuron in the network has the same inputs as before, but with the addition of one single input from its new associated Dendrite Node.

With Dendrite Nodes added, the network once again performs backpropagation learning. However, the error backpropagation is still performed in exactly the same way, which means only through the neurons of the network. The connection weights from the Dendrite Nodes are modified together with the connections from the other presynaptic neurons in the network, but no error is propagated through them. Once again with backpropagation learning the network will reach a learning asymptote. When this occurs another set of Dendrite Nodes is added in the same manner. The one difference being that new Dendrite Nodes for a neuron also receive input from all previous Dendrite Nodes created for that individual neuron, as is done in Cascade Correlation. Adding these artificial Dendrite Nodes to neurons of an established pre-trained network allows the neurons to continue to code for the same feature they were coding for in the original design. However, each additional Dendrite Node allows every neuron in the architecture to reduce their error at coding for that feature.

This achieves the two goals of the project. First the Dendrite Nodes are outside the network. This is because, during backpropagation, the learning error is not passed through them. Gradient descent from the output error only flows through the neuron nodes in the network. Additionally, the Dendrite Nodes are outside the network because the original network layers are fully connected but each Dendrite Node only connects to a single neuron postsynaptically. They are still able to affect individual neurons because, in the forward pass, their output connections are formed to their single associated neuron in addition to

any other Dendrite Nodes connected to the same neuron which were created after them. Second, they are different than the neurons in the way they learn. Rather than learning through gradient descent, they use the Cascade Correlation learning system but use the backpropagated error received by their associated neuron rather than the perceptron output error a traditional Cascade Correlation node would have used. A visualization of this process can be seen in Figure 5.

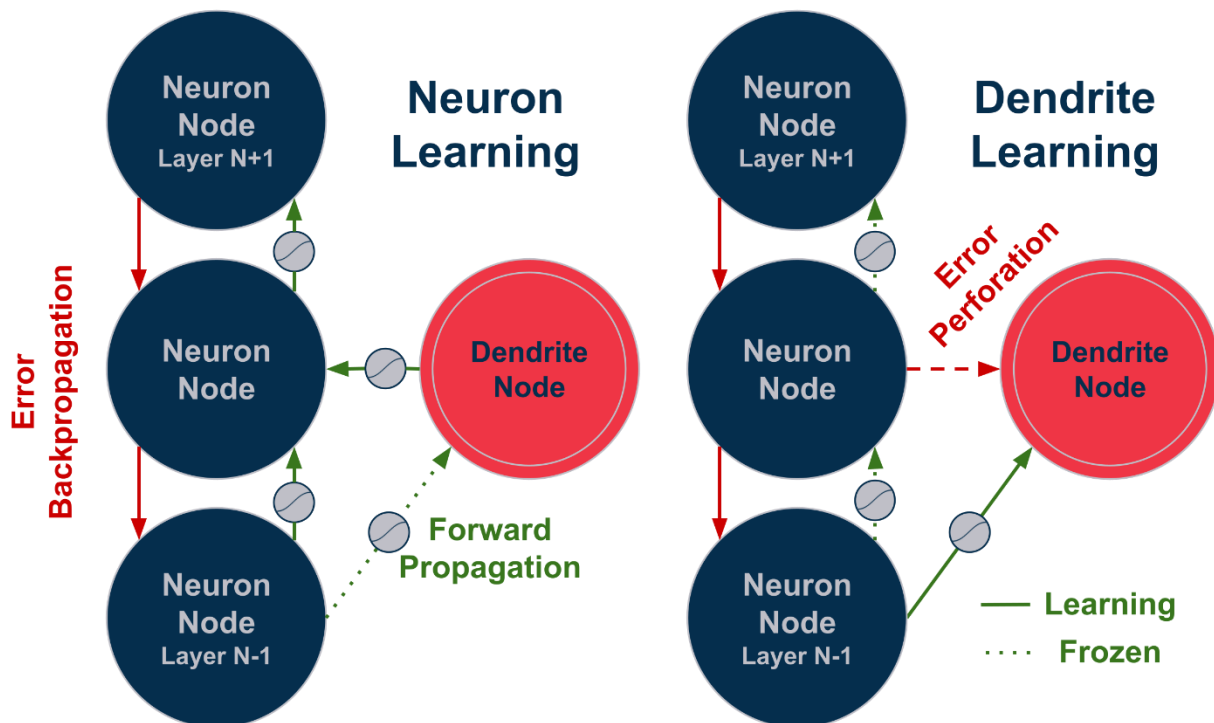


Figure 5. The left image shows the process of neuron learning. Green lines show weights that are being updated. Red lines show the error being backpropagated through the layers. The dotted green line shows weights that are currently frozen and not being adjusted. The right image shows Dendrite Node learning. In this case the dashed red line shows error that is being perforated out of the neuron to the Dendrite Node which then learns its connection weights from the neurons in layer N-1.

Perforated Backpropagation does not only work for multi-layer perceptrons, but also convolutional methods. The behavior is the same where neurons are convolved around their input tensors with the Dendrite Nodes being convolved at the same time. This means every neuron kernel only adds one single input connection per Dendrite Node kernel it adds. That is, with an  $N \times N$  kernel, the postsynaptic neuron will have  $N \times N + 1$  input connections with a single Dendrite Node.

## 5 Results

### 5.1 Baseline test

All experiments were performed with the expected training, validation, and test splits of the datasets. Calculating the scores represented in each graph was performed in the following way. As a network switches back and forth between training neurons and training Dendrites there are a set of epochs and scores associated with each architecture. The training epochs associated with one architecture are defined as a "cycle". The validation score of a cycle is the highest validation score calculated during that cycle.

However, the test score of a cycle is not the highest test score calculated. Instead, the cycle's test score is the test score that was calculated during the same epoch where the highest validation score was calculated. Additionally, the max test score of an experiment is defined as the test score calculated during the epoch where the max validation score was calculated. In each experiment training would take place with these pairs of values being saved until an architecture was created where the highest validation score of that cycle did not improve upon the previous one. This happens in every experiment as Dendrites will always eventually begin to overfit the training data, so early stopping must be performed once this point is reached. Similarly, early stopping is performed during each neuron training cycle by loading the best set of weights of that cycle before adding Dendrites for the next Dendrite cycle.

### Perforated Backpropagation on EMNIST

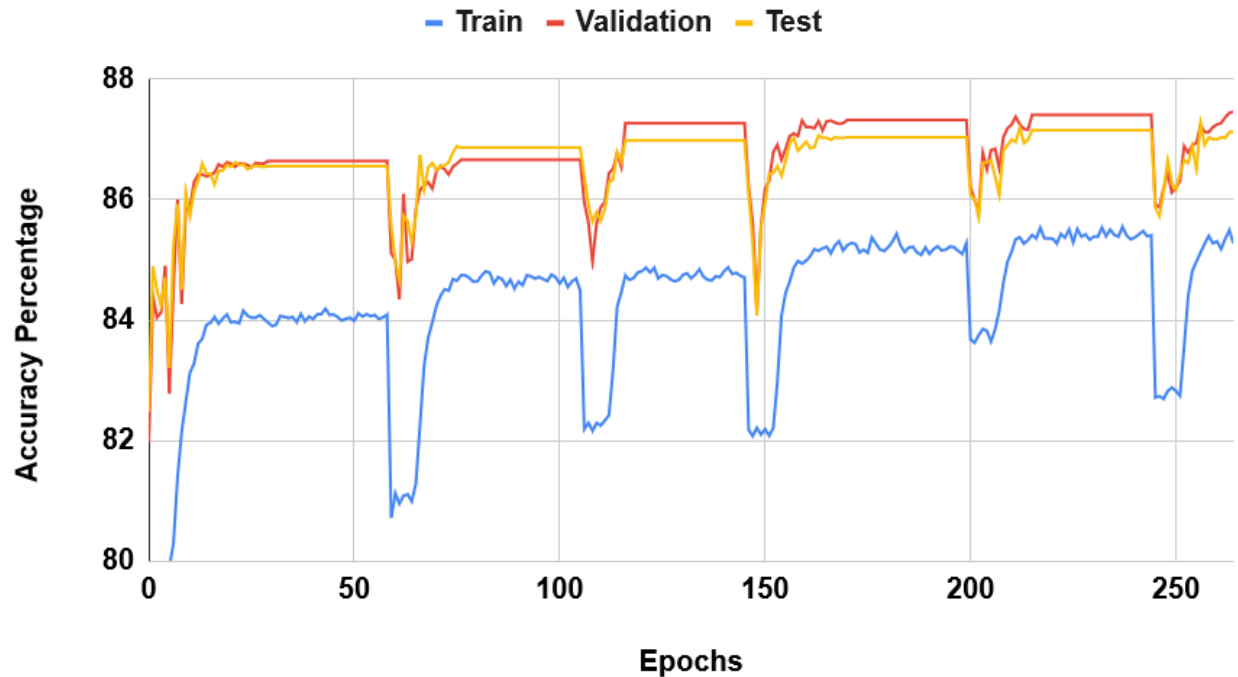


Figure 6. Train, Validation, and Test scores for each epoch of an experiment of Perforated Backpropagation on EMNIST. Epochs in the graph where Test and Validation scores are flatlined over multiple epochs are Dendrite training cycles where the neuron weights are frozen. Training scores of Dendrite cycles still fluctuate because of random shuffling of the dataset. Each dip after a Dendrite cycle occurs as the network adjusts to the new connections. Graph shows Perforated Backpropagation allows scores to go up each cycle after training reaches threshold.

To show a proof of concept, experiments were run to optimize a simple training pipeline. The pipeline chosen was the MNIST (Deng, 2012) example from the official PyTorch GitHub repository (PyTorch, 2024). Because this example started at near perfect accuracy, the MNIST dataset was replaced with the EMNIST Balanced dataset (Cohen, 2017) which contains the same format of images with letters in addition to the original numbers. The network trained has two convolutional layers of widths 32 and 64 leading into two fully connected layers of widths 9216 and 47. All layers use a ReLU activation function, there is a max pooling layer after the second convolution layer, and each fully connected layer is preceded

by a dropout layer (Srivastava, 2014). Training was performed over the EMNIST images with a new cycle beginning when no improvement was seen for 25 epochs in a row. A graph of the exact scores of one of these experiments can be seen in Figure 6.

Additional tests were performed to analyze the importance of each of the components of Perforated Backpropagation. To minimize the impact of random initializations changing the starting scores, Figure 7 shows values for percentage error reduction between the original cycle to the final cycle. For example, the experiment from Figure 6 has a first cycle test accuracy of 86.55% and a final cycle test accuracy of 87.13%, which means the remaining error was reduced by 4.3%. Each experiment was performed seven times. Candlestick values in all charts are minimum, quartile one, quartile three, and maximum.

### Error Reduction on EMNIST

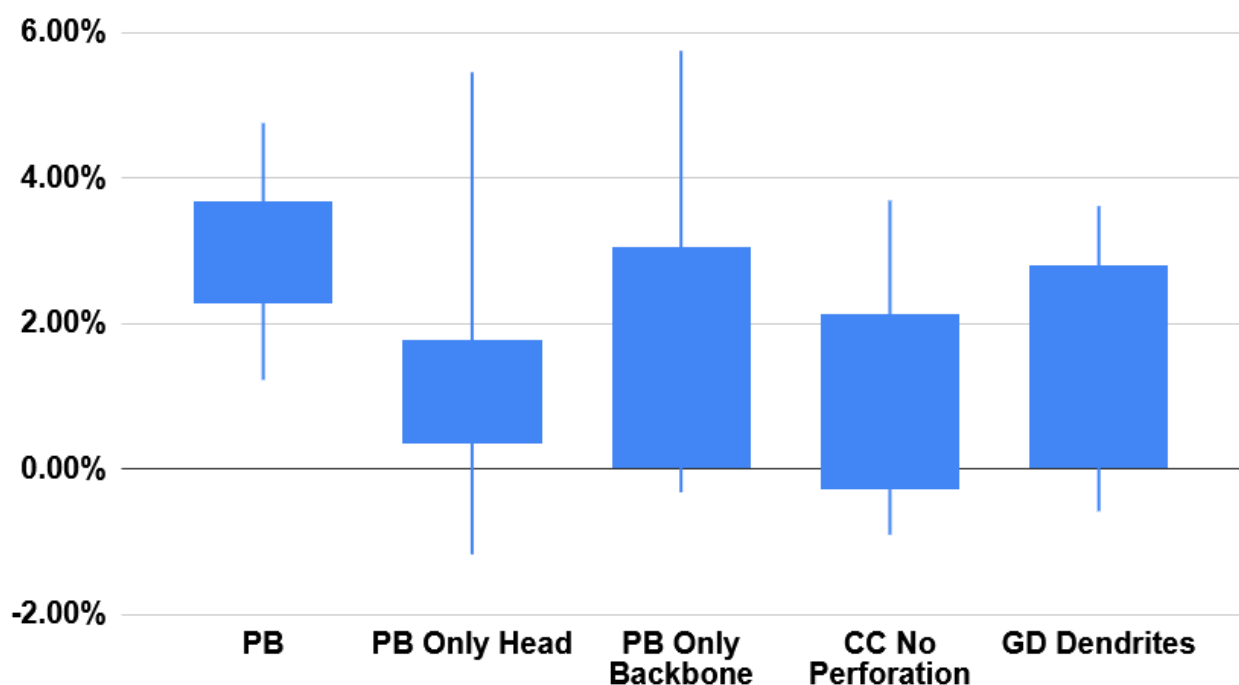


Figure 7. Candlestick charts of test error reduction over seven experiments of each type performed on the EMNIST dataset comparing the test score of the original cycle to the max test score. PB – Perforated Backpropagation. PB Only Head – Only adding Dendrites to the output layer. PB Only Backbone – Adding Dendrites to all layers excluding the output layer. CC No Perforation – Performing Cascade Correlation learning on each layer but also allowing the backpropagation error signal to flow through the new nodes. GD Dendrites – Stopping the error signal from flowing through Dendrites, but training Dendrite weights with gradient descent rather than Cascade Correlation learning.

### 5.2 State-of-the-Art Accuracy Tests

The next experiments performed were on architectures which were state-of-the-art systems in various fields at the times they were published. In each case, the Papers with Code website (PapersWithCode, 2023) was referenced to find networks to improve. This website is a database of datasets. Each dataset contains a list of papers which have been published which utilized that dataset in their experiment. Links

to the open-source repositories are included when code is available. Due to hardware restrictions, datasets were selected where the top systems would fit on a single GPU. All experiments were run on an Ubuntu workstation with a NVIDIA GeForce RTX 2080 GPU and an AMD 9 3900X CPU. In addition, models were chosen that stated they achieved state-of-the-art performance without additional data to train on, other than the official training dataset being competed on. In each case the open-source repository was first downloaded, and the original training protocol was run with recommended settings to determine a baseline. Once this baseline was calculated, Perforated Backpropagation was added into their pipeline and training was run again. Training proceeded in as many cycles as was required until early stopping when validation error stopped improving.

Our most impactful result in this study was on a network architecture called TrimNet (Li, 2021). TrimNet is a graph neural network which utilizes a triplet message mechanism to learn molecular representation. We found TrimNet when it was the state-of-the-art system on the Tox21 dataset from MoloculeNet (Wu Z, 2018). Tox21 is a toxicology dataset which is used to learn the toxicity of various chemical compounds. The dataset contains "dense features" that represent chemical descriptors and "sparse features" that represent chemical substructures. For each compound machine learning systems are expected to predict if the compound will be active or inactive in 12 different toxicological experiments.

### AUC Scores of TrimNet on Tox21 with Perforated Backpropagation

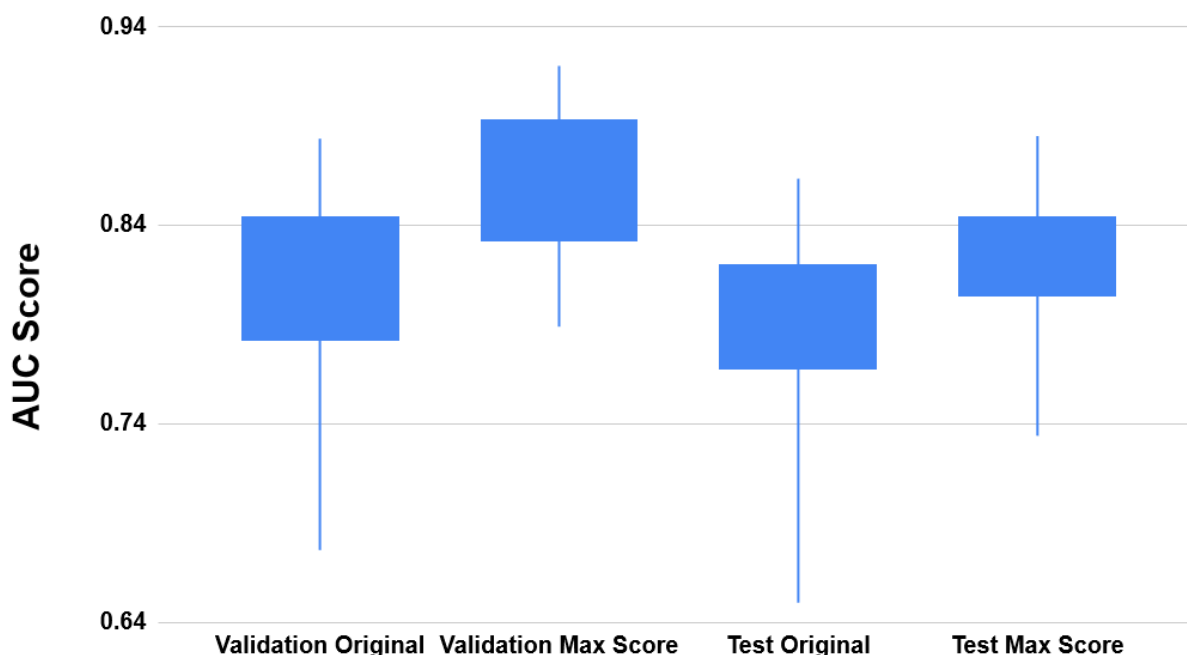


Figure 8. Candlestick chart of AUC scores of our fifty runs of Perforated Backpropagation enhanced TrimNet on the Tox 21 dataset. Results show the max test score has higher values than the original test score.

The open-source code was downloaded and run with default settings to determine a baseline result. Perforated Backpropagation was then added to their system and the experiment was run fifty times with fifty different random initializations. This experiment was run over so many iterations because of the high fluctuation of scores depending on the random seed of the training. Over the experiments, on average, the test AUC score was able to be pushed 13.6% closer to a perfect score of 1.0. The average AUC

score when replicating the original experiment was 0.789. The average test score taken at the point of the highest validation score over a full experiment was 0.822. Training would be run until a cycle where additional Dendrite Nodes started to decrease the validation score. A table showing how many Dendrite Nodes were added to final models can be seen in Table 1. The authors of TrimNet’s published result of 0.860 AUC is above our average, but below our highest test score over the experiments of 0.885. In that particular instance, our random initialization gave us a starting score of 0.807 which meant the error reduction of 40.4% was much better than our average.

Number of Dendrites	0	1	2	3	4	5	6	7	8	9	10	11	12
Experiment Count	2	2	8	11	5	2	7	7	2	2	1	0	1

Table 1. Table showing how many Dendrite Nodes were used for optimization in each of the fifty experiments. Two out of fifty experiments showed Dendrites did not improve upon the original. The maximum added was a single experiment which added twelve.

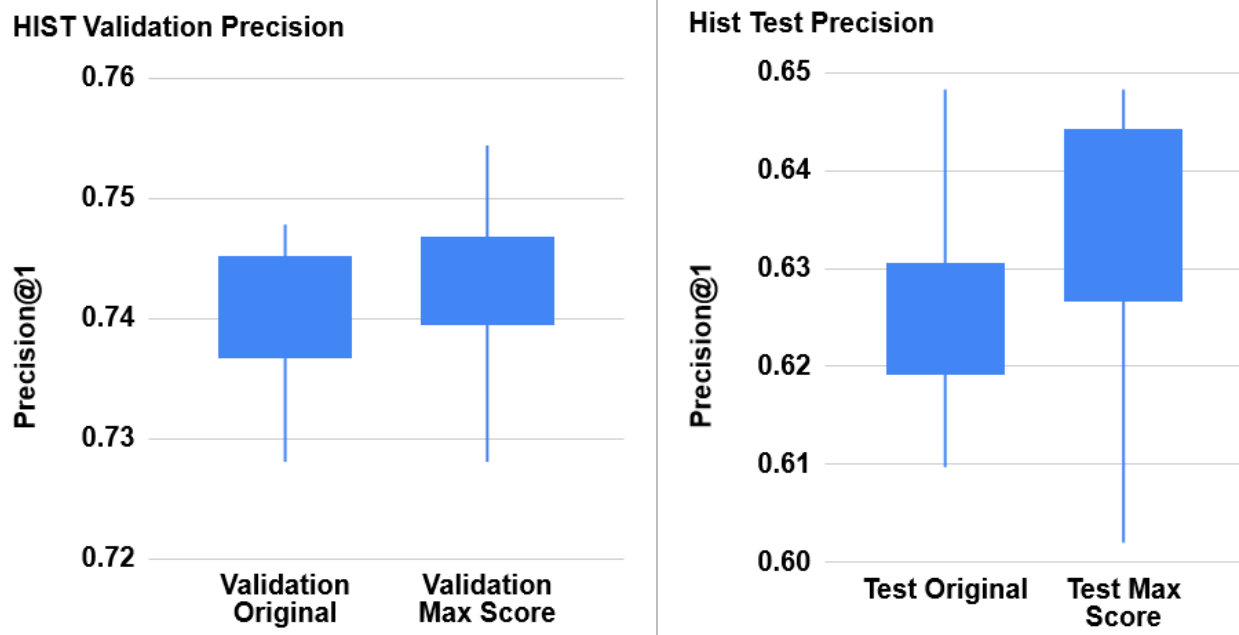


Figure 9. Candlestick chart of validation and test scores of Perforated Backpropagation with HIST on the CSI300 dataset over ten experiments. Results show the max test score has higher values than the original test score.

The second experiment was using Microsoft’s qlib stock trend forecasting repository (qlib, 2023). This repository contains a library of implemented models to train and test on the CSI300 dataset first introduced in (Weiguang Han, 2023). This dataset contains opening price, closing price, highest price, lowest price, volume weighted average price and volume looking back 60 days to construct a 360-dimensional historical stock data as input. Networks train on this data to make decisions about how each stock will perform in the future. The network chosen to work with was the HIST model (Xu, 2021). HIST is a graph-based framework designed to mine the concept-oriented shared information between stocks

while making decisions about each stock. The Precision@1 metric is determined by the percentage of predictions where the top-rated stock has a correct label. Because random predictions would grant a score of 0.5, this was the score used as the baseline for determining our improvements above the original score. For example, our best result raised the original test score from 0.610 to 0.648, granting a 35% increase above 0.5. On average, over 10 experiments with different random initializations an 8.6% improvement was found.

### 5.3 Test Addressing the Increase in Parameters

Each time Dendrites are added to these systems the total number of free parameters of the architecture increases by approximately the total number of parameters of the original model. This is because Dendrites have weighted connections to each of the same input neurons as the neuron they are associated with. A question that this raises, is whether these networks are only improving because the total number of parameters is increasing. To answer that question, an experiment was run to determine the effect of parameter changes by adjusting the number of parameters with Dendrites compared to adjusting the number of parameters via the width of the network. In a network with fully-connected layers, adjusting the number of nodes per layer causes a quadratic change in the number of parameters. Adding Dendrite Nodes causes a linear change. This final experiment is on the PhysioNet Challenge from 2012 (Ikaro Silva, 2012). The PhysioNet dataset is de-identified records of patients in Intensive Care Units. Each record consists of roughly 48 hours of irregularly sampled multivariate time series data used to predict in-hospital mortality of the patients. The system chosen to work with on this dataset is a called Multi Time Attention Network (mTAN) (Satya Narayan Shukla, 2021).

**mTAN Average Scores by Parameter Count**

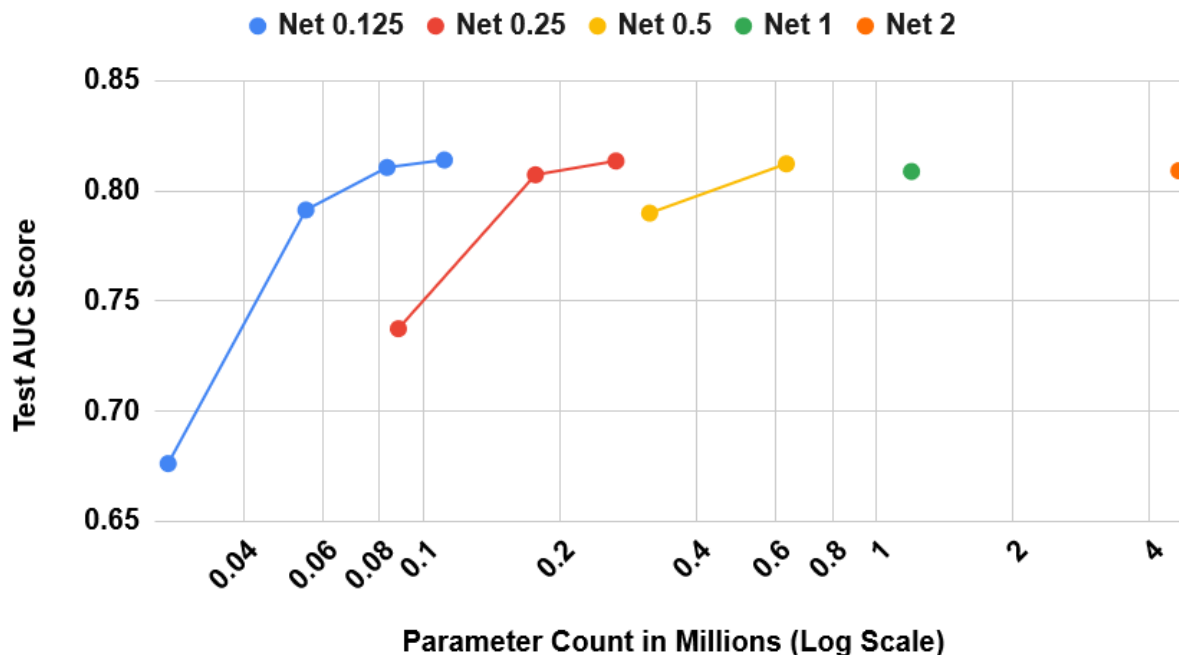


Figure 10. Graph showing total parameter counts compared to average test AUC scores of each network tested over seven experiments. Networks are labeled by their width multiplier. Multiple points show network's initial values as well as subsequent values as

Dendrite Nodes are added. Results show that smaller initial networks with Perforated Backpropagation can create smaller final models without a loss in AUC.

Graphs showing the results over various network widths can be seen in Figure 10 and Figure 11. In Figure 10 the leftmost point represents the original score of that architecture while each subsequent point shows the scores calculated with each additional Dendrite added. Three Dendrites were included for Net 0.125, two for Net 0.25 and one for Net 0.5 because there were rarely improvements with additional Dendrites for each network after that number of cycles. Dendrites were not included for Net 1 and Net 2 because these networks regularly overfit with the addition of the first Dendrite. In these experiments the final Net 0.125 architecture is not only less than a tenth the size of the original, but the average score of Net 0.125 with three dendrites is 2.7% better than Net 1. Hence, our improved performance with Dendrite Nodes cannot be explained by sheer parameter count.

**mTAN Scores by Network Architecture**

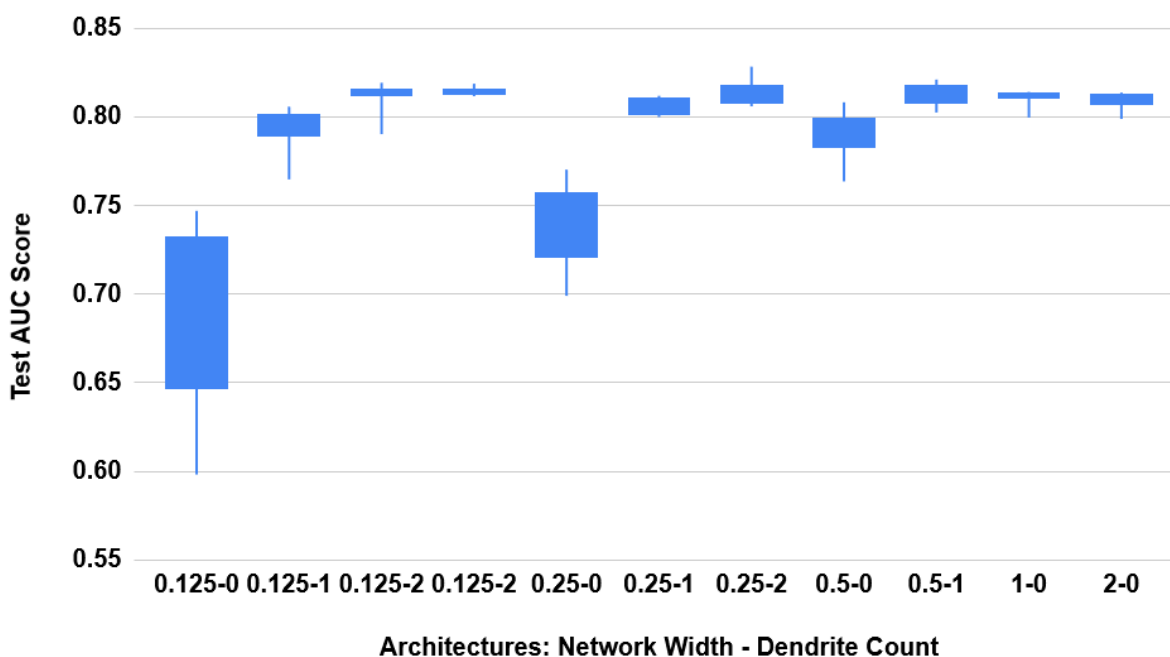


Figure 11. Candlestick chart of each architecture with each number of dendrites over the seven tests at each width corresponding to the points in Figure 10. Results show the variance is values from Figure 10.

In addition to Dendrites impacting parameter counts, the architecture changes also cause increases to both training time and inference time. In this experiment both neuron training cycles and Dendrite training cycles each consisted of 250 epochs. This means that while the original architecture only took 250 epochs in total, Net 0.125 trained for 1750 epochs. With Net 0.125's smaller size the original epochs were faster, but by the later epochs the additional Dendrites cause each training epoch to be slower than the original. The average epoch time and total training times for each network can be seen in Table 2. In addition, Table 2 contains timing information for running inference on each architecture. CPU inference was run a single batch at a time and GPU inference was processed in batches of 100. Inference times are not shown for



Dendrite cycles because inference is not run with networks which have Dendrites that are not connected to neurons.

		<b>Seconds Per Training Epoch</b>	<b>Seconds Per Training Cycle</b>	<b>GPU Inference Inputs / Second</b>	<b>CPU Inference Inputs / Second</b>
<b>Net 0.125</b>	Neuron 1	5.15	1287.28	3026.21	52.12
	Neuron 2	7.59	1897.93	1528.8	30.24
	Neuron 3	10.03	2507.28	1019.72	21.90
	Neuron 4	12.61	3170.31	733.04	16.43
	Dendrite 1	9.43	2357.83		
	Dendrite 2	11.89	2970.20		
	Dendrite 3	14.68	3670.25		
	<b>Total Training Time</b>		<b>17861.08</b>		
<b>Net 0.25</b>	Neuron 1	5.20	1299.20	3059.39	17.52
	Neuron 2	7.59	1897.91	1551.21	25.60
	Neuron 3	9.98	2493.77	1030.59	41.31
	Dendrite 1	9.50	2375.40		
	Dendrite 2	11.89	2972.27		
	<b>Total Training Time</b>		<b>11038.55</b>		
<b>Net 0.5</b>	Neuron 1	5.69	1423.27	2576.98	18.37
	Neuron 2	8.51	2126.69	1293.69	32.26
	Dendrite 1	10.70	2674.49		
	<b>Total Training Time</b>		<b>6224.45</b>		
<b>Net 1</b>	<b>Total Training Time</b>	7.13	<b>1781.96</b>	1783.37	26.46
<b>Net 2</b>	<b>Total Training Time</b>	10.41	<b>2602.96</b>	1177.45	15.90

Table 2. Timing information for training and inference of the mTAN network. Values show the addition of Dendrites has a significant time cost during training and a modest time cost during inference.

In neural network training there is a tradeoff between accuracy and model size. The first two experiments showed that Dendrites can offer increased accuracy at the cost of increased parameters. This experiment shows that Dendrites can also decrease parameters without a loss in accuracy. However, this experiment additionally shows that even with smaller models the current implementation still has a cost in runtime. While inference only sees a modest increase in time to process each input, training time to add Dendrites brings a significant increase in the total time to train a model because of how many additional epochs must be run.

## 5.4 Failure Cases

Additional successful experiments were conducted with other architectures that were not state-of-the-art including DenseNet (Huang, 2017), ResNet (He, 2016), Vision Transformer (Dosovitskiy, 2020), and UNet (Ronneberger, 2015). However, several unsuccessful experiments were performed as well. While Perforated Backpropagation can be added to any PyTorch system, in the capacity that it can run, there are instances where there is not actually any improvement. Tests without success include EfficientDet (Tan, 2020), SpinalNet (Kabir, 2022), Bibert (Qin, 2022), and IGMTF (Xu, 2021). With basic networks which are either linear or convolutional multi-layer perceptrons the system always works. But with skip connections, attention mechanisms, recurrent processing, and other modern deep learning paradigms there are mixed results. Further research still needs to be done to figure out if these architectures are incompatible with Perforated Backpropagation or if it is only the current implementation that is causing problems.

We are optimistic the experiments which fail do so because of the current implementation and not the underlying theory. This is because the correlation scores being learned by Dendrite Nodes are able to be tracked and there is a pattern with successful and unsuccessful experiments. With simple networks, Perforated Backpropagation can be added to a system with only a few lines of code and the Dendrites are consistently able to learn correlation properly. However, with more complicated networks, custom processing functions must be written for what to do when the output of a layer of neurons goes through additional processing before being sent to the neurons of a subsequent layer. The Dendrite Nodes associated with these neurons of the preceding layer prior to this processing are the ones where correlation is frequently not able to be learned. We hope that with further research and by working with the authors of these networks who have a deeper understanding of the processing involved the correct processing functions could be built so Perforated Backpropagation could improve them as well.

In addition to cases where Dendrite Nodes are unable to properly learn a correlation, there are also cases where they do learn a correlation and perform as expected but that leads to overfitting on the training dataset and performing worse on validation and test. In these cases, we do not believe implementation changes could improve results, but these are cases where the model compression ability of Perforated Backpropagation can still be used, such as with the mTAN architecture which did not show improvement with the original network size.

## 6 Possible Biological Implications

Artificial neurons do not mirror the way biological neurons work. But when artificial neurons were first created, they were called neurons because they do share key features of biological neurons. Both combine input from presynaptic connections, the connections have the concept of a connection weight, and the input is passed through a nonlinearity after being combined. As with artificial neurons, the artificial dendrites of Perforated Backpropagation do not mirror the way biological dendrites work. However, it is equally appropriate to call these new nodes artificial dendrites as it is to call the original nodes artificial neurons because they share key features of biological dendrites which are not present in standard artificial neurons.

Two points regard the nodes themselves, while the rest are related to how they learn. First, the original artificial neurons simplified the complexity of a biological neuron's action potential with a simple thresholding function. This later evolved into sigmoid, TanH, ReLu and the other nonlinear functions more commonly found in modern networks. Similarly, these modern functions are used for Dendrite Nodes rather than implementing the full complexity of a biological dendrite's method of nonlinear integration. Despite the simplifications of the true biological mechanics, we argue it is more biologically realistic to

have presynaptic input for an artificial neuron go through additional nonlinear functions via Dendrite Nodes rather than direct presynaptic-postsynaptic connections of traditional artificial neural networks. Second, Dendrite Nodes are closer in architectural design to biological dendrites than they are to biological neurons, because they postsynaptically connect to only a single neuron and other Dendrite Nodes connected to that neuron.

Biology also supports key features of our perforated learning paradigm. Hebbian learning is an accepted system in neuroscience, the idea that when neurons fire together temporally their connection is strengthened. Connection strengths are also modified with other mechanisms, many of which transpire on the postsynaptic side of the synapse. As some examples, postsynaptic cells exhibiting a rise in calcium can lead to increased neurotransmitter receptors at synapses (Gerrow & Triller, 2010). To stabilize Hebbian learning average level of postsynaptic depolarization can decrease synapse strength (Leslie, et al., 2001; Abbott & Nelson, 2000). Postsynaptic synapse size has been shown to be directly related to synapse strength (Meyer, et al., 2014). Protein kinase A, found more concentrated in dendrites than the rest of the neuron, regulates long term potentiation (Zhong, et al., 2009). This idea that so much of synapse strength is modified by the postsynaptic cell is evidence that in biology the post synaptic cell has great influence on the connection strengths it forms to presynaptic neurons. Perforated Backpropagation also has a learning rule where the error calculated at the postsynaptic artificial neuron is the value used to adjust the strengths of all connections into its Dendrite Nodes.

We hope future experiments with this algorithm could also be shown to be more computationally biologically realistic than previous models have been. Much work has been done trying to prove that current instantiations of convolutional neural nets already provide a parallel to the biological brain (Parker & Serre, 2015; Kubilius, et al., 2016; Khaligh-Razavi & Kriegeskorte, 2014). But from our research we have found significant evidence against CNNs learning neurologically relevant representations of the human visual system.

This evidence exists both on the level of individual neurons and networks as a whole. Research has shown that in biology individual neurons are sensitive to non-accidental property (NAP) changes in visual shape processing and therefore are coding for some shape aspect which is be more invariant to metric changes than to NAP changes (Kayaert, et al., 2003). In artificial CNNs, neurons instead learn "uninterpretable solutions" and that the individual top layer units do not code for anything more specific than random linear combinations of top layer units (Goodfellow, et al., 2009; Zeiler & Fergus, 2014). Other neural network papers often show various input images or pixel clusters that most strongly activate neurons to argue that each particular neuron is sensitive to a specific quality and claim it learned something important (Krizhevsky, et al., 2012). One group tested neurons at equal depths, from networks which had been trained and networks which instead had instead been given random weights. Two tests are performed, one on MNIST handwritten digits (LeCun, et al., 1998) and a second on a deeper network against ImageNet images (Deng, et al., 2009). The results show that nodes with random weights show the same type of results that some researchers had claimed proves the neurons learned a useful semantic property contained in the training images. This experiment suggests that, when looking at the calculation of a single individual neuron in that network, a random set of weights could be interpreted to be equally useful as a trained node (Szegedy, et al., 2013).

More evidence that CNNs are not utilizing a neurologically relevant representation comes from viewing networks as a whole, testing what kind of input images can cause a fully trained deep CNN to make mistakes. Adversarial images can confuse artificial neural networks (Goodfellow, et al., 2014). If neurons coded for 3-dimensional structure through a hierarchy of NAPs to geons to objects there would

be little difference in the top layer activations from these minor changes to the input. Results also showed across networks and training sets the same adversarial examples would cause similar mistakes. This suggests networks do learn a consistent representation of the input space (Szegedy, et al., 2013). Other research has shown CNNs even learn certain invariances to changes in input (Zeiler & Fergus, 2014). However, failing on these types of changes provides evidence that the representation and invariances learned are not be the same as in the brain. For many of these examples a human viewer would find the modified images indistinguishable from their originals. One cannot argue that these deep networks are not doing impressive calculations, but this is strong evidence that they are not doing them in neurologically relevant ways. Performing similar experiments with our system is planned for future work.

## 7 Conclusion

This paper has shown a novel extension to artificial neural network learning. Based on biological dendrite's non-linear spiking, new nodes can be added to artificial networks. These nodes are used to modify the way presynaptic input is processed to empower single neurons to have improved performance coding for the same features they were originally. Learning with Perforated Backpropagation, where the error gradient does not pass through every node in the network, these new nodes learn input weights to only help reduce error for a single node in the network. Using this framework our results show whether you are starting with a multilayer perceptron, a deep convolutional neural network, or any deep learning architecture, once error has reached a threshold it can be reduced further by adding Dendrite Nodes. Our framework also allows for significant model compression by starting with a reduced framework and then adding Dendrite Nodes to reach the original accuracy while maintaining a reduced size.

These results are highly encouraging that Dendrite Nodes performing additional non-linear processing for individual neurons in between hidden layers of deep neural networks could be a promising direction to create better machine learning systems. We publish this research with the hope that there will be interest in working with our approach in the deep learning community.

### 7.2 Reproducing Results and Using our System

Instructions to reproduce the results of this paper can be found in our public GitHub repository at <https://github.com/PerforatedAI/PerforatedAI-Examples>. Instructions and best practices to add Perforated Backpropagation to any project built with PyTorch can be found in our second GitHub repository at <https://github.com/PerforatedAI/PerforatedAI-API>.

## References

- Abbott, L. F. & Nelson, S. B., 2000. Synaptic plasticity: taming the beast. *Nature neuroscience*, Volume 3, p. 1178–1183.
- Alexei Baevski, H. Z. A. M. M. A., 2020. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *arxiv*.
- Arce, F. e. a., 2018. Differential evolution training algorithm for dendrite morphological neural networks. *Applied Soft Computing*, Volume 68, pp. 303-313.
- Biederman, I., n.d. *Recognition-by-components: a theory of human image understanding..* 1987: Psychological review 94.2.
- Branco, T. a. M. H., 2010. The single dendritic branch as a fundamental functional unit in the nervous system. *Current opinion in neurobiology* , Volume 20.4, pp. 494-502.

Chavlis, S. a. P. P., 2021. Drawing inspiration from biological dendrites to empower artificial neural networks. *Current opinion in neurobiology*, Volume 70, pp. 1-10.

Ciregan, D., Meier, U. & Schmidhuber, J., 2012. *Multi-column deep neural networks for image classification*. s.l., s.n., p. 3642–3649.

Cohen, G. e. a., 2017. EMNIST: Extending MNIST to handwritten letters. *2017 international joint conference on neural networks (IJCNN)*. IEEE.

Deng, J. et al., 2009. *Imagenet: A large-scale hierarchical image database*. s.l., s.n., p. 248–255.

Deng, L., 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, Volume 29(6), p. 141–142.

Dosovitskiy, A., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, Volume 2010.11929.

Egrioglu, E. E. B. a. M.-Y. C., 2022. Recurrent dendritic neuron model artificial neural network for time series forecasting. *Information Sciences*, Volume 607, pp. 572-584.

Fahlman, S. & Lebiere, C., 1989. The cascade-correlation learning architecture. *Advances in neural information processing systems*, Volume 2.

Gao, S. e. a., 2018. Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. *IEEE transactions on neural networks and learning systems*, Volume 30.2, pp. 601-614.

Gerrow, K. & Triller, A., 2010. Synaptic stability and plasticity in a floating world. *Current opinion in neurobiology*, Volume 20, p. 631–639.

Goodfellow, I. J., Shlens, J. & Szegedy, C., 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Goodfellow, I. et al., 2009. Measuring invariances in deep networks. *Advances in neural information processing systems*, Volume 22.

Guerguiev, J. T. P. L. a. B. A. R., 2017. Towards deep learning with segregated dendrites. *Elife* 6, Volume e22901.

He, K. e. a., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*.

He, K., Zhang, X., Ren, S. & Sun, J., 2016. *Deep residual learning for image recognition*. s.l., s.n., p. 770–778.

Huang, G. e. a., 2017. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Huggingface, 2024. [https://huggingface.co/docs/transformers/model\\_doc/wav2vec2](https://huggingface.co/docs/transformers/model_doc/wav2vec2). [Online].

Ikaro Silva, G. M. S. C. R. M., 2012. Predicting in-hospital mortality of icu patients: The physionet computing in cardiology challenge. *Computingincardiology*, 39:245–248.

Kabir, H. D. e. a., 2022. Spinalnet: Deep neural network with gradual input.. *IEEE Transactions on Artificial Intelligence*.

Kandel, E. R. et al., 2000. Principles of neural science. vol. 4 McGraw-hill. New York.

Kayaert, G., Biederman, I. & Vogels, R., 2003. Shape tuning in macaque inferior temporal cortex. *Journal of Neuroscience*, Volume 23, p. 3016–3027.

Khaligh-Razavi, S.-M. & Kriegeskorte, N., 2014. Deep supervised, but not unsupervised, models may explain IT cortical representation. *PLoS computational biology*, Volume 10, p. e1003915.

Koch, C., Poggio, T. & Torre, V., 1983. Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing.. *Proceedings of the National Academy of Sciences*, Volume 80, p. 2799–2802.

Krizhevsky, A. & H. G., 2009. Learning Multiple Layers of Features from Tiny Images.

Krizhevsky, A., Sutskever, I. & Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, Volume 25.

Kubilius, J., Bracci, S. & Op de Beeck, H. P., 2016. Deep neural networks as a computational model for human shape sensitivity. *PLoS computational biology*, Volume 12, p. e1004896.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Volume 86, p. 2278–2324.

LeCun, Y. e. a., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Volume 86.11, pp. 2278-2324.

Leslie, K. R., Nelson, S. B. & Turrigiano, G. G., 2001. Postsynaptic depolarization scales quantal amplitude in cortical pyramidal neurons. *The Journal of neuroscience*, Volume 21, p. RC170.

Linnainmaa, S., 1970. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors..* s.l.:Diss. Master's Thesis, Univ. Helsinki.

Li, P. e. a., 2021. TrimNet: learning molecular representation from triplet messages for biomedicine. *Briefings in Bioinformatics*.

Liu, Z. e. a., 2024. Kan: Kolmogorov-arnold networks.. *arXiv preprint arXiv:2404.19756*.

Li, X. e. a., 2020. Power-efficient neural network with artificial dendrites. *Nature Nanotechnology*, Volume 15.9, pp. 776-782.

Major, G., Larkum, M. E. & Schiller, J., 2013. Active properties of neocortical pyramidal neuron dendrites. *Annual review of neuroscience*, Volume 36, p. 1–24.

Major, G. et al., 2008. Spatiotemporally graded NMDA spike/plateau potentials in basal dendrites of neocortical pyramidal neurons. *Journal of neurophysiology*, Volume 99, p. 2584–2601.

McCulloch, W. S. a. W. P., 1943. A logical calculus of the ideas immanent in nervous activity.. *The bulletin of mathematical biophysics*, Volume 5, pp. 115-133..

Meir, Y. e. a., 2023. Learning on tree architectures outperforms a convolutional feedforward network. *Scientific Reports* , Volume 13.1.

Meyer, D., Bonhoeffer, T. & Scheuss, V., 2014. Balance and stability of synaptic structures during synaptic plasticity. *Neuron*, Volume 82, p. 430–443.

Norvig, S. R. a. P., 2003. *Artificial Intelligence A Modern Approach, Second Edition*. s.l.:Pearson Education.

PapersWithCode, 2023. *Paperswithcode.com*. [Online].

Parker, S. M. & Serre, T., 2015. Unsupervised invariance learning of transformation sequences in a model of object recognition yields selectivity for non-accidental properties. *Frontiers in computational neuroscience*, Volume 9, p. 115.

Parmar, N. e. a., 2018.. Image transformer. Volume International conference on machine learning.

PyTorch, 2024. *PyTorch Examples*. [Online]  
Available at: <https://github.com/pytorch/examples/tree/main/mnist>

Qin, H. e. a., 2022. Bibert: Accurate fully binarized bert.. *arXiv preprint arXiv:2203.06390*.

qlib, M., 2023. <https://qlib.readthedocs.io/en/latest/>. [Online].

Rall, W., 1964. Theoretical significance of dendritic trees for neuronal input-output relations. *Neural theory and modeling*.

Ritter, G. X. L. I. a. G. U., 2003. Morphological perceptrons with dendritic structure. *The 12th IEEE International Conference on Fuzzy Systems*, Volume 2.

Ronneberger, O. P. F. a. T. B., 2015. U-net: Convolutional networks for biomedical image segmentation. *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015*.

Rosenblatt, F., 1958. *The perceptron: a probabilistic model for information storage and organization in the brain*. s.l.:Psychological review 65.6 386..

Russakovsky, O. et al., 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, Volume 115, p. 211–252.

Satya Narayan Shukla, B. M. M., 2021. Multi-Time Attention Networks for Irregularly Sampled Time Series. *Arxiv*.

Shu-wen Yang, P.-H. C. Y.-S. C. C.-I. J. L. K. L. Y. Y. L. A. T. L. J. S. X. C. G.-T. L. T.-H. H. W.-C. T. K.-t. L. D.-R. L. Z. H. S. D. S.-W. L. S. W., 2021. SUPERB: Speech processing Universal PERFORMANCE Benchmark. *Arxiv*.

Sossa, H. a. E. G., 2014. Efficient training for dendrite morphological neural networks. *Neurocomputing*, Volume 131, pp. 132-142..

Srivastava, N. e. a., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, Volume 15.1, pp. 1929-1958.

Stuart, G. & Spruston, N., 1998. Determinants of voltage attenuation in neocortical pyramidal neuron dendrites. *Journal of Neuroscience*, Volume 18, p. 3501–3510.

Szegedy, C. et al., 2015. *Going deeper with convolutions*. s.l., s.n., p. 1–9.

Szegedy, C. et al., 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tan, M. R. P. a. Q. V. L., 2020. Efficientdet: Scalable and efficient object detection.. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Vinyals O, B. S. K. M. e. a., 2016. Order matters: sequence to sequence for sets. In: International Conference on Learning Representations. *ICLR Press*.

Weiguang Han, B. Z. Q. X. M. P. Y. L. J. H., 2023. Select and Trade: Towards Unified Pair Trading with Hierarchical Reinforcement Learning. *Arxiv*.

Widrow, B. & Lehr, M. A., 1990. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, Volume 78, p. 1415–1442.

Wu Z, R. B. F. E. e. a., 2018. MoleculeNet: a benchmark for molecular machine learning. *Chem Sci*.

Xu, W. e. a., 2021. Hist: A graph-based framework for stock trend forecasting via mining concept-oriented shared information. *Arxiv*.

Xu, W. e. a., 2021. Instance-wise graph-based framework for multivariate time series forecasting.. *arXiv preprint arXiv:2109.06489*.

Zeiler, M. D. & Fergus, R., 2014. *Visualizing and understanding convolutional networks*. s.l., s.n., p. 818–833.

Zhong, H. et al., 2009. Subcellular dynamics of type II PKA in neurons. *Neuron*, Volume 62, p. 363–374.